Complexity of a Problem Concerning Reset Words for Eulerian Binary Automata

Vojtěch Vorel

Charles University in Prague, Czech Republic
vorel@ktiml.mff.cuni.cz

Abstract. A word is called a reset word for a deterministic finite automaton if it maps all states of the automaton to one state. Deciding about the existence of a reset word of given length for a given automaton is known to be a NP-complete problem. We prove that it remains NP-complete even if restricted on Eulerian automata over the binary alphabet, as it has been conjectured by Martyugin (2011).

1 Introduction

A *deterministic finite automaton* is a triple $A = (Q, X, \delta)$, where Q and X are finite sets and δ is an arbitrary mapping $Q \times X \to Q$. Elements of Q are called *states*, X is the *alphabet*. The *transition function* δ can be naturally extended to $Q \times X^\star \to Q$, still denoted by δ. We extend it also by defining $\delta(S, w) = \{\delta(s, w) \mid s \in S, w \in X^\star\}$ for each $S \subseteq Q$.

For a given automaton $A = (Q, X, \delta)$, we call $w \in X^\star$ a *reset word* if $|\delta(Q, w)| = 1$. If such a word exists, we call the automaton *synchronizing*. Note that each word having a reset word as a factor is also a reset word.

The *Černý conjecture*, a longstanding open problem, claims that each synchronizing automaton has a reset word of length $(|Q| - 1)^2$. However, there are many weaker results in this field, see e.g. [6,7] for recent ones.

Various computational problems arises from study of synchronization:

- *Given an automaton, decide if it is synchronizing.* Relatively simple algorithm which could be traced back to [1] works in polynomial time.
- *Given a synchronizing automaton and a number d, decide if d is the length of shortest reset words.* This has been shown to be both NP-hard [2] and coNP-hard. More precisely, it is DP-complete [5].
- *Given a synchronizing automaton and a number d, decide if there exists a reset word of length d.* This problem is of our interest. Lying in NP, it is not so computationally hard as the previous problem. However, it is proven to be NP-complete [2]. Following the notation of [4], we call it Syn. Assuming that $\mathcal{M}$ is a class of automata and membership in $\mathcal{M}$ is polynomially decidable, we define a restricted problem:

  Syn($\mathcal{M}$)
  Input: synchronizing automaton $A = ([n], X, \delta) \in \mathcal{M}$, $d \in \mathbb{N}$
  Output: does A have a reset word of length d?

A.-H. Dediu et al. (Eds.): LATA 2014, LNCS 8370, pp. 576–587, 2014.

An automaton $A = (Q, X, \delta)$ is *Eulerian* if

$$\sum_{x \in X} |\{r \in Q \mid \delta(r, x) = q\}| = |X|$$

for each $q \in Q$. Informally, there must be exactly $|X|$ transitions incoming to each state. An automaton is *binary* if $|X| = 2$. The classes of Eulerian and binary automata are denoted by $\mathcal{EU}$ and $\mathcal{AL}_2$ respectively.

Previous results about various restrictions of SYN can be found in [2,3,4]. Some of these problems turned out to be polynomially solvable, others are NP-complete. In [4] Martyugin conjectured that $\text{SYN}(\mathcal{EU} \cap \mathcal{AL}_2)$ is NP-complete. This conjecture is confirmed in the rest of the present paper.

2 Main Result

Proof Outline. We prove the NP-completeness of $\text{SYN}(\mathcal{EU} \cap \mathcal{AL}_2)$ by polynomial reduction from 3-SAT. So, for arbitrary propositional formula ϕ in 3-CNF we construct an Eulerian binary automaton A and a number d such that

$$\phi \text{ is satisfiable} \Leftrightarrow A \text{ has a reset word of length } d. \tag{1}$$

For the rest of the paper we fix a formula $\phi = \bigwedge_{i=1}^{m} \bigvee_{\lambda \in C_i} \lambda$ on n variables where each C_i is a three-element set of literals, i.e. subset of

$$L_\phi = \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}.$$

We index the literals by the mapping κ defined by

$$\kappa : x_1 \mapsto 0, \ldots, x_n \mapsto n-1, \neg x_1 \mapsto n, \ldots, \neg x_n \mapsto 2n-1.$$

Let $A = (Q, X, \delta)$, $X = \{a, b\}$. Because the structure of the automaton A will be very heterogenous, we use an unusual method of description. The basic principles of the method are:

- We describe the automaton A via labeled directed multigraph G, representing the automaton in a standard way: edges of G are labeled by single letters a and b and carry the structure of the function δ. Paths in G are *labeled* by words from $\{a, b\}^\star$.
- There is a collection of labeled directed multigraphs called *templates*. The graph G is one of them. Another template is `SINGLE`, which consists of one vertex and no edges.
- Each template $\mathtt{T} \neq \mathtt{SINGLE}$ is a disjoint union through a set $\text{PARTS}_{\mathtt{T}}$ of its proper subgraphs (the *parts* of `T`), extended by a set of additional edges (the *links* of `T`). Each $H \in \text{PARTS}_{\mathtt{T}}$ is isomorphic to some template `U`. We say that H *is of type* `U`.

– Let q be a vertex of a template $\mathtt{T}$, lying in subgraph $H \in \mathrm{PARTS}_{\mathtt{T}}$ which is of type $\mathtt{U}$ via vertex mapping $\rho : H \to \mathtt{U}$. The *local adress* $\mathrm{adr}_{\mathtt{T}}(q)$ is a finite string of identifiers separated by "|". It is defined inductively by

$$\mathrm{adr}_{\mathtt{T}}(q) = \begin{cases} H \mid \mathrm{adr}_{\mathtt{U}}\, \rho(q) & \text{if } \mathtt{U} \neq \mathtt{SINGLE} \\ H & \text{if } \mathtt{U} = \mathtt{SINGLE}. \end{cases}$$

The string $\mathrm{adr}_G(q)$ is used as regular vertex identifier.

Having a word $w \in X^\star$, we denote a t-th letter of w by w_t and define the set $S_t = \delta(Q, w_1 \ldots w_t)$ of *active states at time t*. Whenever we depict a graph, a solid arrow stands for the label a and a dotted arrow stands for the label b.

Description of the Graph G

Let us define all the templates and informally comment on their purpose. Figure 1 defines the template `ABS`, which does not depend on the formula ϕ.

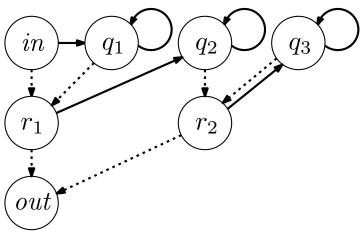

Fig. 1. Template `ABS`

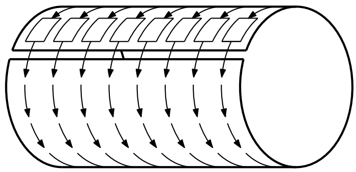

Fig. 2. A barrier of `ABS` parts

The state *out* of a part of type `ABS` is always inactive after application of a word of length at least 2 which does not contain b^2 as a factor. This allows us to ensure the existence of a relatively short reset word. Actually, large areas of the graph (namely the `CLAUSE(...)` parts) have roughly the shape depicted in Figure 2, a cylindrical structure with a horizontal barrier of `ABS` parts. If we use a sufficiently long word with no occurence of b^2, the edges outgoing from the `ABS` parts are never used and almost all states become inactive.

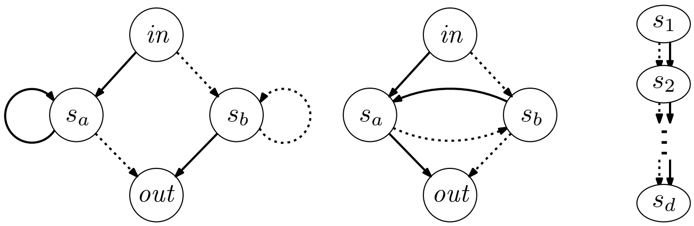

Fig. 3. Templates `CCA`, `CCI` and `PIPE`(d)

Figure 3 defines simple templates CCA, CCI and PIPE(d) for each $d \geq 1$. If we secure constant activity of the *in* state, the activity of the *out* state depends exactly on the last two letters applied. In the case of CCA it gets inactive if and only if the two letters were equal. In the case of CCI it works oppositely, equal letters correspond to active *out* state. One of the key ideas of the entire construction is the following. Let there be a subgraph of the form

$$\begin{array}{c} \text{part of type } \mathtt{PIPE}(d) \\ \downarrow a,b \\ \text{part of type } \mathtt{CCA} \text{ or } \mathtt{CCI} \\ \downarrow a,b \\ \text{part of type } \mathtt{PIPE}(d). \end{array} \tag{2}$$

Before the synchronization process starts, all the states are active. As soon as the second letter of an input word is applied, the activity of the *out* state starts to depend on the last two letters and the pipe below keeps a record of its previous activity. We say that a part H of type PIPE(d) *records* a sequence $B_1 \dots B_d \in \{\mathbf{0}, \mathbf{1}\}^d$ *at time* t, if it holds that

$$B_k = \mathbf{1} \Leftrightarrow H|s_k \notin S_t.$$

In order to continue with defining templates, let us define a set M_ϕ containing all literals from L_ϕ and some auxiliary symbols:

$$M_\phi = L_\phi \cup \{y_1, \dots, y_n\} \cup \{z_1, \dots, z_n\} \cup \{q, q', r, r'\}.$$

We index the $4n+4$ members of M_ϕ by the following mapping μ:

$\nu \in M_\phi$	q	r	y_1	x_1	y_2	x_2	...	y_n	x_n
$\mu(\nu)$	1	2	3	4	5	6	...	$2n+1$	$2n+2$

$\nu \in M_\phi$	q'	r'	z_1	$\neg x_1$	z_2	$\neg x_2$	...	z_n	$\neg x_n$
$\mu(\nu)$	$2n+3$	$2n+4$	$2n+5$	$2n+6$	$2n+7$	$2n+8$	...	$4n+3$	$4n+4$

The inverse mapping is denoted by μ'. For each $\lambda \in L_\phi$ we define templates INC(λ) and NOTINC(λ), both consisting of $12n+12$ SINGLE parts identified by elements of $\{1,2,3\} \times M_\phi$. As depicted by Figure 4, the links of INC(λ) are:

$$(1,\nu) \xrightarrow{a} \begin{cases} (2,r) & \text{if } \nu = \lambda \text{ or } \nu = r \\ (2,\nu) & \text{otherwise} \end{cases} \qquad (1,\nu) \xrightarrow{b} \begin{cases} (2,\lambda) & \text{if } \nu = \lambda \text{ or } \nu = r \\ (2,\nu) & \text{otherwise} \end{cases}$$

$$(2,\nu) \xrightarrow{a} \begin{cases} (3,q) & \text{if } \nu = \lambda \text{ or } \nu = q \\ (3,\lambda) & \text{if } \nu = r \\ (3,\nu) & \text{otherwise} \end{cases} \qquad (2,\nu) \xrightarrow{b} \begin{cases} (3,r) & \text{if } \nu = \lambda \text{ or } \nu = q \\ (3,\lambda) & \text{if } \nu = r \\ (3,\nu) & \text{otherwise} \end{cases}$$

Note that we use the same identifier for an one-vertex subgraph and for its vertex. The structure of NOTINC(λ) is clear from Figure 5.

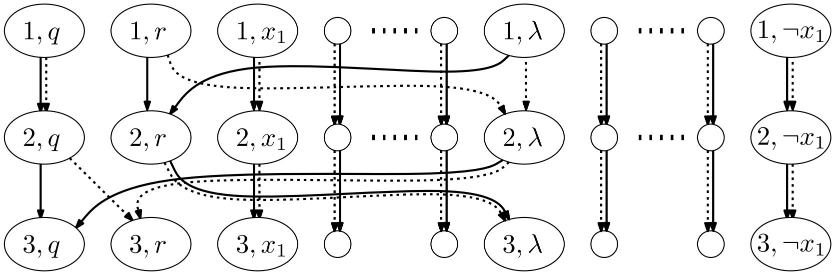

Fig. 4. Template `INC`(λ)

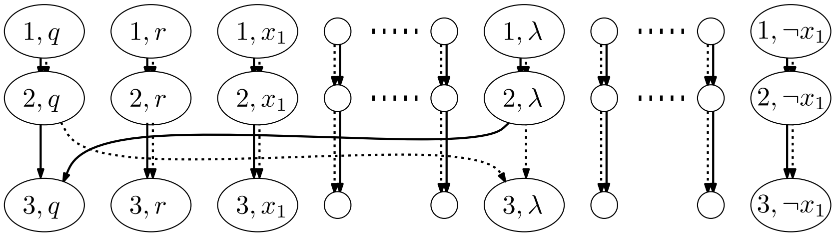

Fig. 5. Template `NOTINC`(λ)

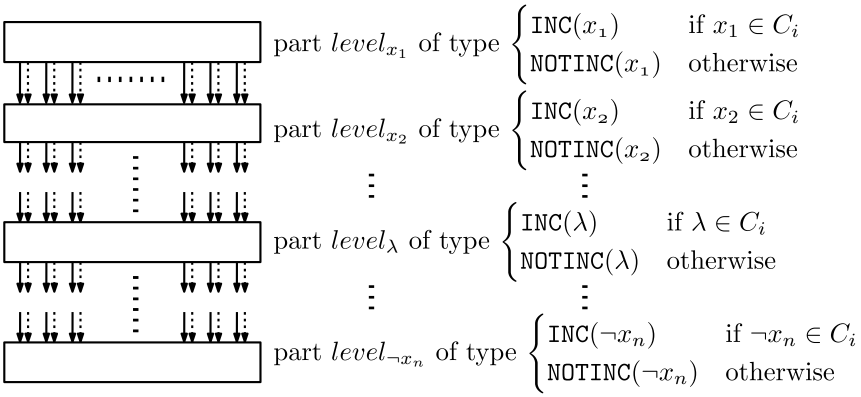

Fig. 6. Template `TESTER`

The key property of such templates comes to light when we need to apply some two-letter word in order to make the state $(3, \lambda)$ inactive assuming $(1, r)$ inactive. If also $(1, \lambda)$ is initially inactive, we can use the word a^2 in both templates. If it is active (which corresponds to the idea of unsatisfied literal λ), we discover the difference between the two templates: The word a^2 works if the type is `NOTINC(`λ`)`, but fails in the case of `INC(`λ`)`. Such failure corresponds to the idea of unsatisfied literal λ occuring in certain clause of ϕ.

For each clause (each $i \in \{1, \ldots, m\}$) we define a template `TESTER(`i`)`. It consists of $2n$ serially linked parts, namely $level_\lambda$ for each $\lambda \in L_\phi$, each of type `INC(`λ`)` or `NOTINC(`λ`)`. The particular type of each $level_\lambda$ depends on the clause C_i as seen in Figure 6, so exactly three of them are always of type `INC(...)`. If the corresponding clause is unsatisfied, each of its three literals is unsatisfied, which causes three *failures* within the levels. Three failures imply at least three occurences of b, which turns up to be too much for a reset word of certain length to exist. Clearly we still need some additional mechanisms to realize this vague vision.

Figure 7 defines templates `FORCER` and `LIMITER`. The idea of template `FORCER` is simple. Imagine a situation when $q_{1,0}$ or $r_{1,0}$ is active and we need to deactivate the entire forcer by a word of length at most $2n + 3$. Any use of b would cause an unbearable delay, so if such a word exists, it starts by a^{2n+2}.

The idea of `LIMITER` is similar, but we tolerate some occurences of b here, namely two of them. This works if we assume $s_{1,0}$ active and it is neccesary to deactivate the entire limiter by a word of length at most $6n + 1$.

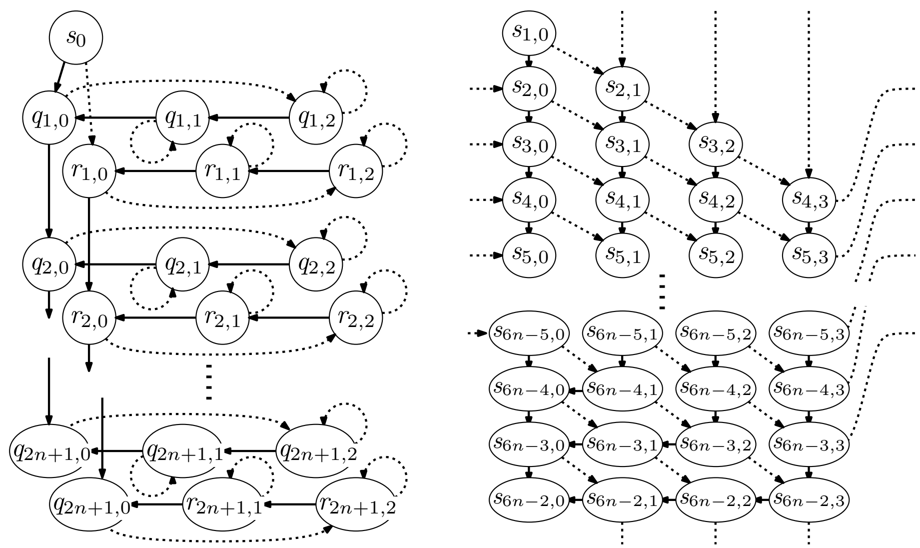

Fig. 7. Templates `FORCER` and `LIMITER` respectively

We also need a template $\mathtt{PIPES}(d,k)$ for each $d, k \geq 1$. It consists just of k parallel pipes of length d. Namely there is a SINGLE part $s_{d',k'}$ for each $d' \leq d$, $k' \leq k$ and all the edges are of the form $s_{d',k'} \longrightarrow s_{d'+1,k'}$.

The most complex templates are $\mathtt{CLAUSE}(i)$ for each $i \in \{1, \dots, m\}$. Denote

$$\alpha_i = (i-1)(12n-2),$$
$$\beta_i = (m-i)(12n-2).$$

As shown in Figure 8, $\mathtt{CLAUSE}(i)$ consists of the following parts:

- Parts $sp_1, \dots, sp_{4n+6}$ of type SINGLE.
- Parts $abs_1, \dots, abs_{4n+6}$ of type ABS. All the template have a shape similar to Figure 2, including the barrier of ABS parts.
- Parts $pipe_2$, $pipe_3$, $pipe_4$ of types $\mathtt{PIPE}(2n-1)$ and $pipe_6$, $pipe_7$ of types $\mathtt{PIPE}(2n+2)$.
- Parts cca and cci of types CCA and CCI respectively. Together with the pipes above they realize the idea described in (2). As they form two constellations which work simultaneously, the parts $pipe_6$ and $pipe_7$ typically record mutually inverse sequences. We interpret them as an assignment of the variables $x_1, \dots, x_n$. Such assignment is then processed by the tester.
- A part ν of type SINGLE for each $\nu \in M_\phi$.
- The part *tester* of type $\mathtt{TESTER}(i)$.
- A part $\overline{\lambda}$ of type SINGLE for each $\lambda \in L_\phi$. While describing the templates $\mathtt{INC}(\lambda)$ and $\mathtt{NOTINC}(\lambda)$ we claimed that in certain case there arises a need to make the state $(3, \lambda)$ inactive. This happens when the border of inactive area moves down through the tester levels. The point is that any word of length $6n$ deactivates the entire tester, but we need to ensure that some tester columns, namely the $\kappa(\lambda)$-th for each $\lambda \in L_\phi$, are deactivated one step earlier. If some of them is still active just before the deactivation of tester finishes, the state $\overline{\lambda}$ becomes active, which slows down the sychronizing process.
- Parts $pipes_1$, $pipes_2$ and $pipes_3$ of types $\mathtt{PIPES}(\alpha_i, 4n+4)$, $\mathtt{PIPES}(6n-2, 4n+4)$ and $\mathtt{PIPES}(\beta_i, 4n+4)$ respectively. There are multiple clauses in ϕ, but multiple testers cannot work in parallel. That is why each of them is padded by a passive $\mathtt{PIPES}(\dots)$ part of size depending on particular i. If $\alpha_i = 0$ or $\beta_i = 0$, the corresponding PIPES part is not present in cl_i.
- Parts $pipe_1$, $pipe_5$, $pipe_8$, $pipe_9$ of types $\mathtt{PIPE}(12mn+4n-2m+6)$, $\mathtt{PIPE}(4)$, $\mathtt{PIPE}(\alpha_i+6n-1)$, $\mathtt{PIPE}(\beta_i)$ respectively.
- The part *forcer* of type FORCER. This part guarantees that only the letter a is used in certain segment of the word w. This is nessesary for the data produced by cca and cci to safely leave the parts $pipe_3$, $pipe_4$ and line up in the states of the form ν for $\nu \in M_\phi$, from where they shift to the tester.
- The part *limiter* of type LIMITER. This part guarantees that the letter b occurs at most twice when the border of inactive area passes through the tester. Because each usatisfied literal from the clause requests an occurence of b, only a satisfied clause meets all the conditions for a reset word of certain length to exist.

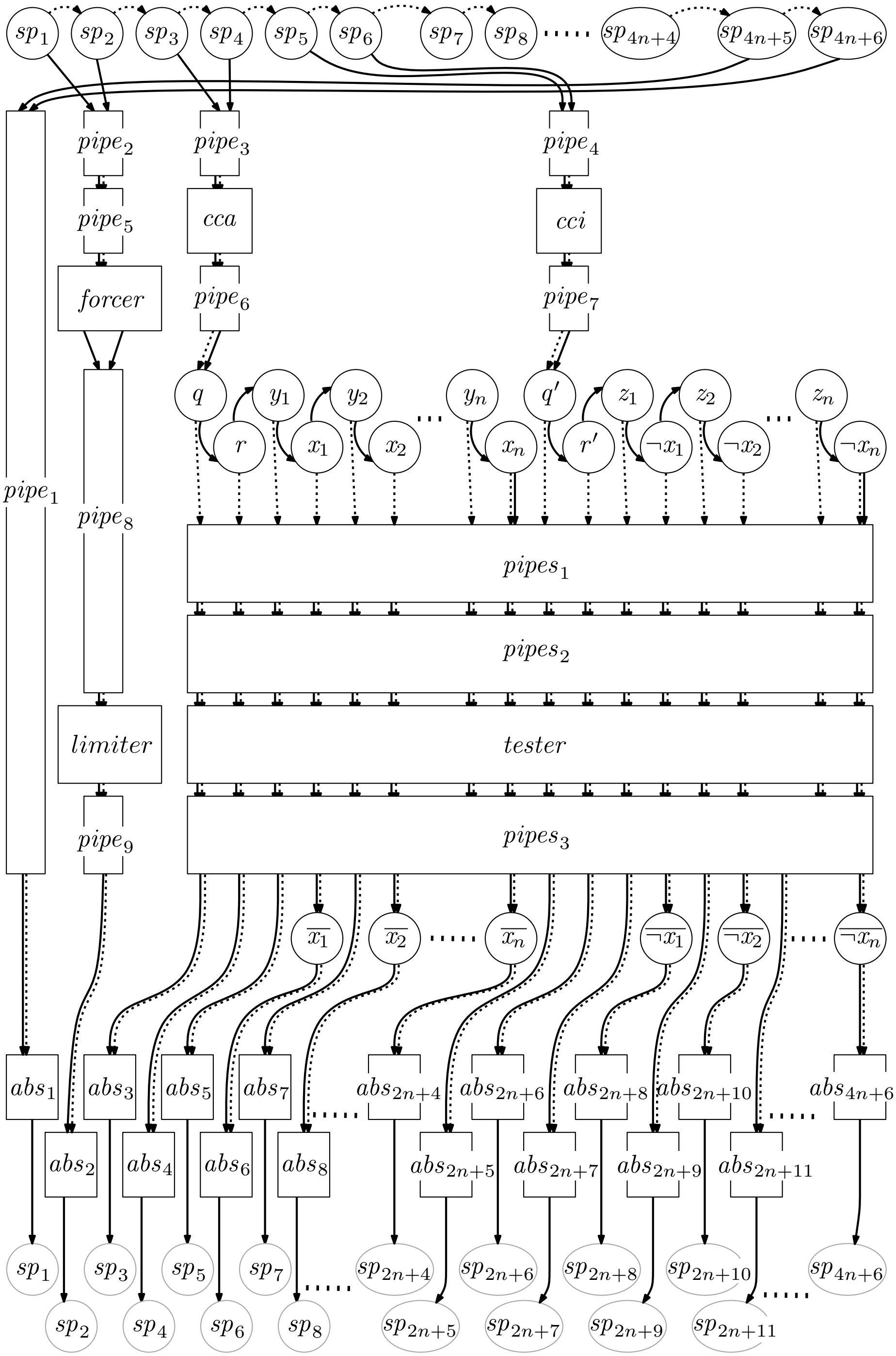

Fig. 8. Template `CLAUSE(`i`)`

Links of $\mathtt{CLAUSE}(i)$, which are not clear from the Figure 8 are

$$\nu \xrightarrow{a} \begin{cases} pipes_1|s_{1,\mu(\nu)} & \text{if } \nu = \neg x_n \\ \mu'(\mu(\nu)+1) & \text{otherwise} \end{cases} \qquad \nu \xrightarrow{b} pipes_1|s_{1,\mu(\nu)}$$

for each $\nu \in M_\phi \setminus \{\neg x_n\}$ and

$$pipes_3|s_{\beta_i,k} \xrightarrow{a,b} \begin{cases} \overline{\mu'(k)} & \text{if } \mu'(k) \in L_\phi \\ abs_{k+2}|in & \text{otherwise} \end{cases} \qquad \overline{\lambda} \xrightarrow{a,b} abs_{\mu(\lambda)+2}|in$$

for each $k \in \{1,\ldots,4n+4\}$, $\lambda \in L_\phi$.

We are ready to form the whole graph G, see Figure 9. For each $i,k \in \{1,\ldots m\}$ there are parts cl_k, abs_k of types `CLAUSE(i)` and `ABS` respectively and q_k, r_k, r'_k, s_1, s_2 of type `SINGLE`. The edge incoming to a cl_i part ends in $cl_i|sp_1$, the outgoing one starts in $cl_i|sp_{4n+6}$. When no states outside `ABS` parts are active within each `CLAUSE(...)` part and no *out*, r_1 nor r_2 state is active in any `ABS` part, the word b^2ab^{4n+m+7} takes all active states to s_2 and completes the sychronization. Graph G does not fully represent the automaton A yet, because there are

- $8mn+4m$ vertices with only one outgoing edge, namely $cl_i|abs_k|out$ and sp_l for each $i \in \{1,\ldots,m\}, k \in \{1,\ldots,4n+6\}, l \in \{7,\ldots,4n+4\}$,
- $8mn+4m$ vertices with only one incoming edge: $cl_i|\nu$ and $cl_i|pipes_1|(1,\nu')$ for each $i \in \{1,\ldots,m\}, \nu \in M_\phi \setminus \{q,q'\}, \nu' \in M_\phi \setminus \{x_n, \neg x_n\}$.

But we do not need to specify the missing edges exactly, let us just say that they somehow connect the relevant states and the automaton A is complete. Let us set $d = 12mn + 8n - m + 18$ and prove that the equivalence (1) holds.

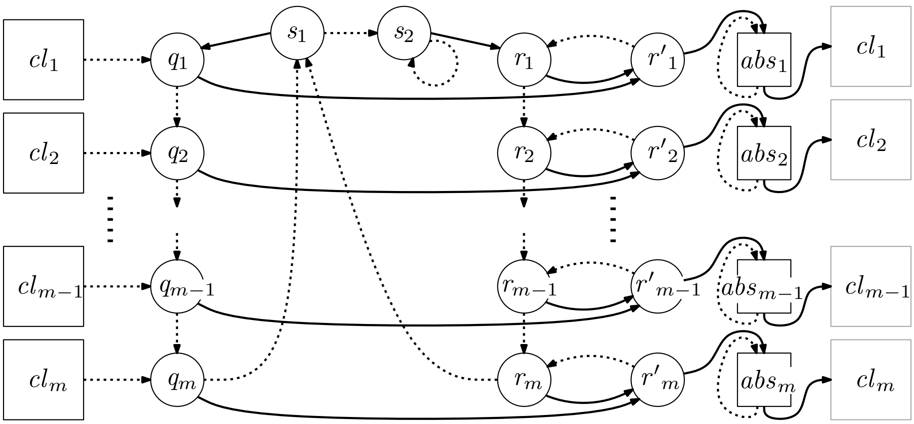

Fig. 9. The graph G

From an Assignment to a Word. At first let us suppose that there is an assignment $\xi_1, \dots, \xi_n \in \{\mathbf{0}, \mathbf{1}\}$ of the variables $x_1, \dots, x_n$ (respectively) satisfying the formula ϕ and prove that the automaton A has a reset word w of length d.

For each $j \in \{1, \dots, n\}$ we denote

$$\sigma_j = \begin{cases} a & \text{if } \xi_j = \mathbf{1} \\ b & \text{if } \xi_j = \mathbf{0} \end{cases}$$

and for each $i \in \{1, \dots, m\}$ we choose a satisfied literal $\overline{\lambda}_i$ from C_i. We set

$$w = a^2 \left(\sigma_n a\right) \left(\sigma_{n-1} a\right) \dots \left(\sigma_1 a\right) aba^{2n+3} b \left(a^{6n-2} v_1\right) \dots \left(a^{6n-2} v_m\right) b^2 ab^{4n+m+7},$$

where for each $i \in \{1, \dots, m\}$ we use the word

$$v_i = u_{i,x_1} \dots u_{i,x_n} u_{i,\neg x_1} \dots u_{i,\neg x_n},$$

denoting

$$u_{i,\lambda} = \begin{cases} a^3 & \text{if } \lambda = \overline{\lambda}_i \text{ or } \lambda \notin C_i \\ ba^2 & \text{if } \lambda \neq \overline{\lambda}_i \text{ and } \lambda \in C_i \end{cases}$$

for each $\lambda \in L_\phi$. We see that $|v_i| = 6n$ and therefore

$$|w| = 4n + 8 + m\left(12n - 2\right) + 4n + m + 10 = 12mn + 8n - m + 18 = d.$$

Let us denote

$$\gamma = 12mn + 4n - 2m + 9.$$

Because the first occurence of b^2 in w starts by γ-th letter, we have:

Lemma 1. *No state of a form $cl_{\dots}|abs_{\dots}|out$ or $abs_{\dots}|out$ lies in any of the sets $S_2, \dots, S_\gamma$.*

Let us fix an arbitrary $i \in \{1, \dots, m\}$ and describe a growing area of inactive states within cl_i. The following claims follows directly from the definition of w. Note that the claim 7 relies on the fact that b occurs only twice in v_i.

Lemma 2

1. *No state of the form $sp_{\dots}$ lies in any of the sets $S_2, \dots, S_\gamma$.*
2. *No state from $pipe_2$ or $pipe_3$ or $pipe_4$ lies in any of the sets $S_{2n+1}, \dots, S_\gamma$.*
3. *No state from cca or cci or $pipe_5$ lies in any of the sets $S_{2n+5}, \dots, S_\gamma$.*
4. *No state from $pipe_6$ or $pipe_7$ or $forcer$ lies in any of the sets $S_{4n+7}, \dots, S_\gamma$.*
5. *No state ν for $\nu \in M_\phi$ lies in any of the sets $S_{4n+8}, \dots, S_\gamma$.*
6. *No state from $pipes_1$ or $pipes_2$ or $pipe_8$ lies in any of the sets $S_{10n+\alpha_i+6}, \dots$ $\dots, S_\gamma$.*
7. *No state from limiter or tester lies in any of the sets $S_{16n+\alpha_i+6}, \dots, S_\gamma$*
8. *No state from $pipe_1$ or $pipe_9$ or $pipes_3$ lies in any of the sets $S_{\gamma-1}, S_\gamma$.*

For each $\lambda \in L_\phi$ we ensure by the word $u_{i,\lambda}$ that the $\kappa\left(\lambda\right)$-th tester column is deactivated in advance, namely at time $t = 16n + \alpha_i + 5$. The advance allows the following key claim to hold true.

Lemma 3. *No state $cl_i|\overline{\lambda}$ for $\lambda \in L_\phi$ lies in any of the sets $S_{\gamma-1}, S_\gamma$.*

We see that within cl_i only states from the `ABS` parts can lie in $S_{\gamma-1}$. Since $w_{\gamma-2}w_{\gamma-1} = a^2$, no state r_1, r_2 or *out* from any `ABS` part lies in $S_{\gamma-1}$. Now we easily check that all the states possibly present in $S_{\gamma-1}$ are mapped to s_2 by the word $w_\gamma \dots w_d = b^2ab^{4n+m+7}$.

From a Word to an Assignment. Since now we suppose that there is a reset word w of length $d = 12mn + 8n - m + 18$. The following lemma is not hard to verify.

Lemma 4

1. *Up to labeling there is unique pair of paths having length at most $d-2$, leading from $cl_1|pipe_1|s_1$ and $cl_2|pipe_1|s_1$ respectively to a common end. They are of length $d-2$ and meet in s_2.*
2. *The word w starts by a^2.*

The second claim implies that for each $i \in \{1, \dots, m\}$ it holds that $cl_i|pipe_1|s_1 \in S_2$, so it follows that

$$\delta(Q, w) = \{s_2\}.$$

Let us denote $\overline{d} = 12mn + 4n - 2m + 11$ and $\overline{w} = w_1 \dots w_{\overline{d}}$. The following lemma holds, because no edges labelled by a are available for final segments of the paths described in the first claim of Lemma 4.

Lemma 5

1. *The word w can be written as $w = \overline{w}b^{4n+m+7}$ for some word $\overline{w}$.*
2. *For any $t \geq \overline{d}$, no state from any $cl_{\dots}$ part lie in S_t, except for the $sp_{\dots}$ states.*

The next lemma is based on properties of the parts $cl_{\dots}|forcer$ but to prove that no more a follows the enforced factor a^{2n+1} we also need to observe that each $cl_{\dots}|cca|out$ or each $cl_{\dots}|cci|out$ lies in S_{2n+4}.

Lemma 6. *The word $\overline{w}$ starts by $\overline{u}a^{2n+1}b$ for some $\overline{u}$ of length $2n+6$.*

Now we are able to write the word $\overline{w}$ as

$$\overline{w} = \overline{u}a^{2n+1}b\,(\overline{v}_1 v'_1 c_1) \dots (\overline{v}_m v'_m c_m)\, w_{\overline{d}-2} w_{\overline{d}-1} w_{\overline{d}},$$

where $|\overline{v}_k| = 6n-2$, $|v'_k| = 6n-1$ and $|c_k| = 1$ for each k and denote $d_i = 10n + \alpha_i + 6$. At time $2n+5$ the parts $cl_{\dots}|pipe_6$ and $cl_{\dots}|pipe_7$ record mutually inverse sequences. Because there is the factor a^{2n+1} after $\overline{u}$, at time d_i we find the information pushed to the first rows of testers:

Lemma 7. *For each $i \in \{1, \dots, m\}$, $j \in \{1, \dots, n\}$ it holds that*

$$cl_i|tester|level_{x_1}|\,(1, x_j) \in S_{d_i} \Leftrightarrow$$
$$cl_i|tester|level_{x_1}|\,(1, \neg x_j) \notin S_{d_i} \Leftrightarrow w_{2n-2j+2} \neq w_{2n-2j+3}.$$

Let us define the assignment $\xi_1, \ldots, \xi_n \in \{\mathbf{0}, \mathbf{1}\}$. By Proposition 7 the definition is correct and does not depend on i:

$$\xi_j = \begin{cases} \mathbf{1} & \text{if } cl_i|tester|level_{x_1}|\,(1, x_j) \notin S_{d_i} \\ \mathbf{0} & \text{if } cl_i|tester|level_{x_1}|\,(1, \neg x_j) \notin S_{d_i}. \end{cases}$$

The following lemma holds due to $cl_{\ldots}|limiter$ parts.

Lemma 8. *For each $i \in \{1, \ldots, m\}$ there are at most two occurences of b in the word v'_i.*

Now we choose any $i \in \{1, \ldots, m\}$ and prove that the assignment $\xi_1, \ldots, \xi_n$ satisfies the clause $\bigvee_{\lambda \in C_i} \lambda$. Let $p \in \{0, 1, 2, 3\}$ denote the number of unsatisfied literals in C_i.

As we claimed before, all tester columns corresponding to any $\lambda \in L_\phi$ have to be deactivated earlier than other columns. Namely, if $cl_i|tester|level_{x_1}|\,(1, \lambda)$ is active at time d_i, which happens if and only if λ is not satisfied by $\xi_1, \ldots, \xi_n$, the word $v'_i c_i$ must not map it to $cl_i|pipes_3|s_{1,\mu(\lambda)}$. If $cl_i|tester|level_\lambda$ is of type `INC(`λ`)`, the only way to ensure this is to use the letter b when the border of inactive area lies at the first row of $cl_i|tester|level_\lambda$. Thus each unsitisfied $\lambda \in C_i$ implies an occurence of b in corresponding segment of v'_i:

Lemma 9. *There are at least p occurences of the letter b in the word v'_i.*

By Lemma 8 there are at most two occurences of b in v'_i, so we get $p \leq 2$ and there is at least one satisfied literal in C_i.

References

1. Černý, J.: Poznámka k homogénnym experimentom s konečnými automatmi. Matematicko-fyzikálny časopis 14(3), 208–216 (1964)
2. Eppstein, D.: Reset sequences for monotonic automata. SIAM J. Comput. 19(3), 500–510 (1990)
3. Martyugin, P.: Complexity of problems concerning reset words for some partial cases of automata. Acta Cybern. 19(2), 517–536 (2009)
4. Martyugin, P.: Complexity of problems concerning reset words for cyclic and eulerian automata. In: Bouchou-Markhoff, B., Caron, P., Champarnaud, J.-M., Maurel, D. (eds.) CIAA 2011. LNCS, vol. 6807, pp. 238–249. Springer, Heidelberg (2011)
5. Olschewski, J., Ummels, M.: The complexity of finding reset words in finite automata. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 568–579. Springer, Heidelberg (2010)
6. Steinberg, B.: The Černý conjecture for one-cluster automata with prime length cycle. Theoret. Comput. Sci. 412(39), 5487–5491 (2011)
7. Trahtman, A.N.: Modifying the upper bound on the length of minimal synchronizing word. In: Owe, O., Steffen, M., Telle, J.A. (eds.) FCT 2011. LNCS, vol. 6914, pp. 173–180. Springer, Heidelberg (2011)